

# **Μια Διαδικτυακή εφαρμογή χρονοπρογραμματισμού και διαχείρισης της υπηρεσίας φύλαξης του Πανεπιστημίου Κρήτης**

Από Ευστράτιος Ταξιάρχης Σικέλης  
ΑΜ: 3151



Ιούλιος 2020  
Επιβλέπων καθηγητής: Μάριος Πιτικάκης  
Επόπτης Καθηγητής: Ιωάννης Τζιτζικας

Τμήμα Επιστήμης Υπολογιστών Πανεπιστημίου Κρήτης

## ΠΕΡΙΕΧΟΜΕΝΑ

1.Εισαγωγή.....	4
1.1 Γενική Περιγραφή.....	5
1.2 Δυνατότητες χρηστών και συστήματος.....	6
1.3. Απαιτήσεις Συστήματος.....	6
2.Οθόνες Εφαρμογής και Σενάρια Χρήσης	
2.1 Log in Screen/ Οθόνη εισόδου.....	7
2.2. Κύρια σελίδα / Σελίδα Εβδομαδιαίου Προγράμματος.....	8
2.2.1. Popup Σελίδα	
2.2.1.1. Εμφάνιση πληροφοριών τρέχοντος event.....	11
2.2.1.2. Δημιουργία event.....	12
2.2.1.3. Διαγραφή επιλεγμένου event.....	13
2.3. Σελίδα αίτησης και απάντησης αιτήσεων αδειών / Ιστορικό.....	13
2.4. Σελίδα Δημιουργίας Εβδομαδιαίου προγράμματος.....	15
2.5. Σελίδα ελέγχου μηνιαίου συνόλου βαρδιών.....	16
2.5.1. Σελίδα ελέγχου μηνιαίου συνόλου βαρδιών.....	17
3. Εργαλεία Κατασκευής Εφαρμογής / Συστήματος.....	18
3.1. ZK Studio - ZK framework.....	18
3.2. Maven.....	19
3.3. MySQL.....	19
3.4. Hibernate ORM .....	19
3.5. Quartz Scheduler.....	19
3.6. Jetty Server.....	19
3.7. OptaPlanner.....	20
4. Backend Αρχιτεκτονική Συστήματος.....	21
4.1. Σύστημα Login.....	22
4.2. Σύστημα Calendar.....	23
4.2.1. Παρουσίαση των Event.....	23
4.2.2. Λειτουργία του Calendar Popup.....	24
4.3. Σύστημα Request Vacation and Vacation History.....	26
4.3.1 Ορισμός Listbox και απάντηση αιτήσεων αδειών.....	26
4.3.2 Κατοχύρωση αίτησης άδειας.....	27
4.4. Σύστημα Ιστορικού Βαρδιών.....	28

4.4.1. Αποθήκευση βαρδιών ημέρας και εμφάνιση συνολικού αριθμού βαρδιών του κάθε φύλακα.....	28
4.4.2. Εμφάνιση ιστορικού συνολικών βαρδιών παλαιότερων μηνών και εμφάνιση γραφημάτων.....	29
4.5. Σύστημα Δημιουργίας προγράμματος φυλάκων.....	31
4.5.1. Αυτόματος χωρισμός φυλάκων.....	32
4.5.2. Ανάθεση φυλάκων με ευθύνη του admin.....	34
4.5.3. Δημιουργία προγράμματος.....	35
4.5.3.1. Δημιουργία seed προγράμματος.....	36
4.5.3.2. Αντιγραφή για μορφοποίηση του seed.....	36
4.5.3.3. Αφαίρεση επαναλαμβανόμενων καταχωρίσεων εργαζομένων στην ίδια βάρδια.....	37
4.5.3.4. Έλεγχος βραδινών βαρδιών.....	37
4.5.3.5. Γέμισμα άδειων βαρδιών.....	38
4.5.3.6. Έλεγχος αδειών εργαζομένων.....	39
5. Αλλαγές, Βελτιώσεις, Επιπλέον features για μελλοντικό development .....	41
5.1. Συμπεριφορά χρονοπρογραμματιστή σε ακραίες περιπτώσεις.....	42
6. References.....	43

## **1. Εισαγωγή**

Ο στόχος της εργασίας είναι ο σχεδιασμός και ανάπτυξη ενός ολοκληρωμένου web-based συστήματος διαχείρισης και υποστήριξης της υπηρεσίας φύλαξης του Πανεπιστημίου Κρήτης στο Ηράκλειο.

Παρακάτω αναφέρονται όλες οι απαιτήσεις και λειτουργίες για τον σωστό χρονοπρογραμματισμό των υπαλλήλων και για την ανακοίνωση και παρακολούθηση του προγράμματος φύλαξης των κτιρίων του ΠΚ.

### **1.1. Γενική περιγραφή**

Το σύστημα θα χρησιμοποιείται από 2 κατηγορίες χρηστών. Ανάλογα με την κατηγορία τους οι χρήστες έχουν τις κατάλληλες διεπαφές για την διεκπεραίωση των αναγκών τους.

Οι χρήστες χωρίζονται σε:

1. admins
2. standard employees

## **1.2. Δυνατότητες χρηστών και συστήματος.**

1. Οι admins έχουν την δυνατότητα δημιουργίας και διαμόρφωσης του εβδομαδιαίου προγράμματος. Μπορούν να παρακολουθούν τις αιτήσεις για άδειες από τους εργαζομένους και το σύνολο των μηνιαίων συμπληρωμένων ωρών εργασίας του κάθε εργαζόμενου. Επιπλέον έχουν την δυνατότητα παρακολούθησης μηνιαίων στατιστικών στοιχείων.
2. Οι απλοί εργαζόμενοι έχουν την δυνατότητα παρακολούθησης των δικών τους εβδομαδιαίων βάρδιών στο πρόγραμμα. Και την δημιουργία αιτήσεων για άδεια.

## **1.3. Απαιτήσεις Συστήματος**

Ο χρονοπρογραμματισμός των εργαζομένων πρέπει να γίνεται σύμφωνα με κανόνες που ορίστηκαν από τις υπηρεσίες του ΠΚ.

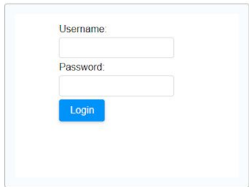
1. Δύο κτίρια προς χρονοπρογραμματισμό. Πανεπιστημιούπολη Βουτών και Κνωσού.
2. Ο κάθε φύλακας έχει μία βάρδια την ημέρα.
3. Υπάρχουν τρεις βάρδιες την ημέρα. 06:00-14:00, 14:00-22:00, 22:00-06:00.
4. Υπάρχουν 13 φύλακες προς χρονοπρογραμματισμό.
5. Δικαιούνται 2 ημέρες ανάπαυσης.
6. Εργάζονται σε όλες τις βάρδιες από Δευτέρα έως Κυριακή και σε όλους τους χώρους.
7. Η κάθε βάρδια μπορεί να έχει από 0 έως 3 φύλακες. Αλλά μπορούν να οριστούν ελάχιστες υποχρεωτικές ανάγκες φύλαξης για συγκεκριμένες βάρδιες.
8. Να υπάρχει τουλάχιστον ανάπαυση 8 ωρών για φύλακες που εργάστηκαν στην βάρδια 3 (νυχτερινή).
9. Οι βάρδιες 1&2 από Δευτέρα-Σάββατο περιλαμβάνεται στον βασικό μισθό. Το ημερομίσθιο σε νυχτερινές βάρδιες και βάρδιες σε αργίες και Κυριακές είναι προσαυξανόμενο.
10. Κάθε χρήστης έχει δικό του username/password.
11. Να καταχωρηθούν επίσημες αργίες του δημοσίου.
12. Να καταχωρούνται οι άδειες των φυλάκων κάθε εβδομάδας και να συμπεριλαμβάνονται στην δημιουργία του προγράμματος.
13. Να καταχωρείται ο αριθμός των φυλάκων που θα εργαστούν κάθε βάρδια.
14. Δημιουργία εβδομαδιαίου προγράμματος λαμβάνοντας υπόψη περιορισμούς της εβδομάδας και δυνατότητα τροποποίησης από τον admin.
15. Υπολογισμός ωρών εργασίας και προσαύξησης για κάθε φύλακα και υπολογισμός των συνολικών ωρών εργασίας του φύλακα.

- 16. Δυνατότητα εξαγωγής στοιχείων σε αρχείο.
- 17. Δυνατότητα εξαγωγής στατιστικών στοιχείων.

## 2. Οθόνες Εφαρμογής και Σενάρια Χρήσης

### 2.1. Log in Screen/ Οθόνη εισόδου

Η πρώτη σελίδα που βλέπει ο χρήστης είναι η σελίδα εισόδου όπου ο χρήστης εισάγει τα προσωπικά του δεδομένα για να κάνει login. Ο χρήστης πρέπει να εισάγει σωστά στοιχεία εισόδου. Εμφανίζεται μήνυμα λάθους σε άλλη περίπτωση.



The screenshot displays a login interface within a light blue rectangular frame. Inside this frame, there is a smaller white rectangular area containing the login form. The form consists of two text input fields: the top one is labeled 'Username:' and the bottom one is labeled 'Password:'. Below these fields is a blue button with the text 'Login' in white. The entire login form is centered within the larger frame.

## 2.2. Κύρια σελίδα / Σελίδα Εβδομαδιαίου Προγράμματος.

Αυτή η σελίδα διαφέρει ανάλογα με το είδος του χρήστη που έκανε login. Εάν ο χρήστης έκανε είσοδο ως admin η σελίδα του περιέχει μερικές επιπλέον επιλογές.

Admin login:

The screenshot shows the 'Request Vacation' interface for an admin user. At the top, there are three buttons: 'Request Vacation' (active), 'Create Schedule', and 'Billing'. Below these are tabs for 'Day', 'Week', and 'Month', with 'Day' selected. Navigation arrows are on the right. The calendar grid shows dates from Sunday, June 31, to Saturday, July 4. The grid is populated with numerous vacation requests, each represented by a colored bar (yellow, green, or black) with a text label indicating the start and end times (e.g., 'Πανεπιστημιακή Βουλή 22:00-06:00'). Some requests are highlighted with a yellow background, and others with a black background. The interface is designed to allow the admin to manage and view vacation requests for the entire team.

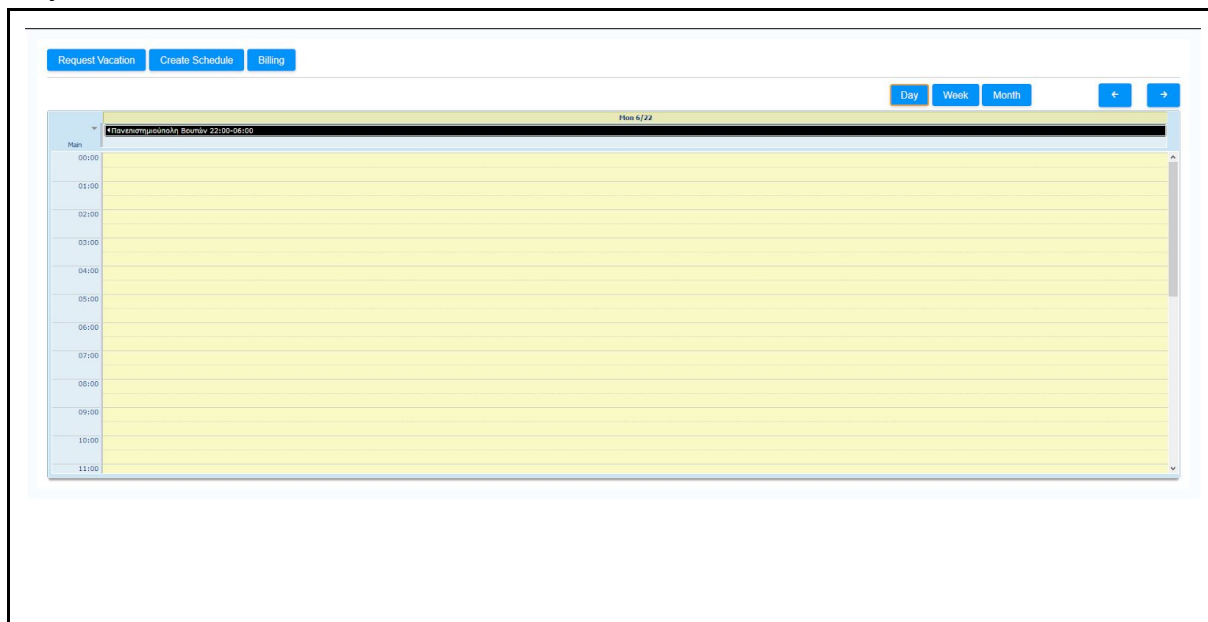
Standard Employee login:

The screenshot shows the 'Request Vacation' interface for a standard employee. It features a 'Request Vacation' button at the top left. The calendar view is similar to the admin view, with tabs for 'Day', 'Week', and 'Month', and 'Day' selected. The calendar grid shows dates from Sunday, June 31, to Saturday, July 4. The grid displays a few vacation requests as colored bars with labels (e.g., 'Πανεπιστημιακή Βουλή 14:00-22:00'). The interface is simpler than the admin version, focusing on the employee's own vacation requests.

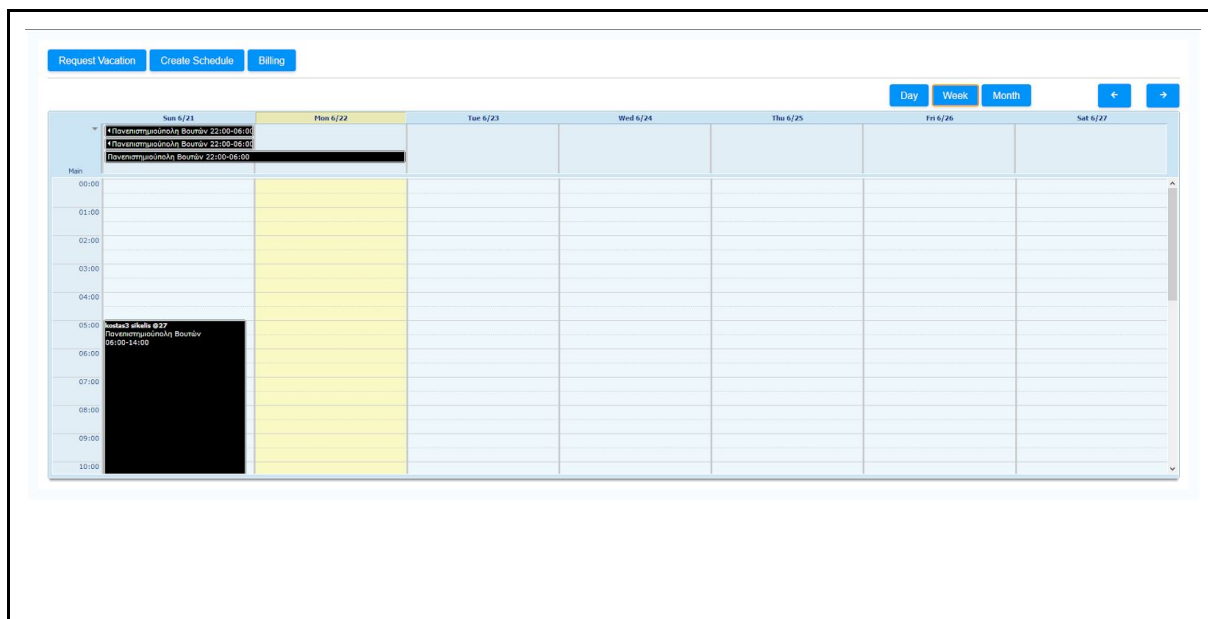


Και στις 2 περιπτώσεις ο χρήστης μπορεί να δει το πρόγραμμα της τρέχουσας ημέρας αλλά και τα προγράμματα των προηγούμενων εβδομάδων. Μπορεί ο χρήστης να αλλάξει τον τρέχων μήνα στο ημερολόγιο και μπορεί επιπλέον να αλλάξει το model του ημερολογίου. Η default μορφή του ημερολογίου είναι αυτή που φαίνεται στις προηγούμενες εικόνες. Ο χρήστης πατώντας τα κουμπιά Day και Week μπορεί να αλλάξει την μορφή.

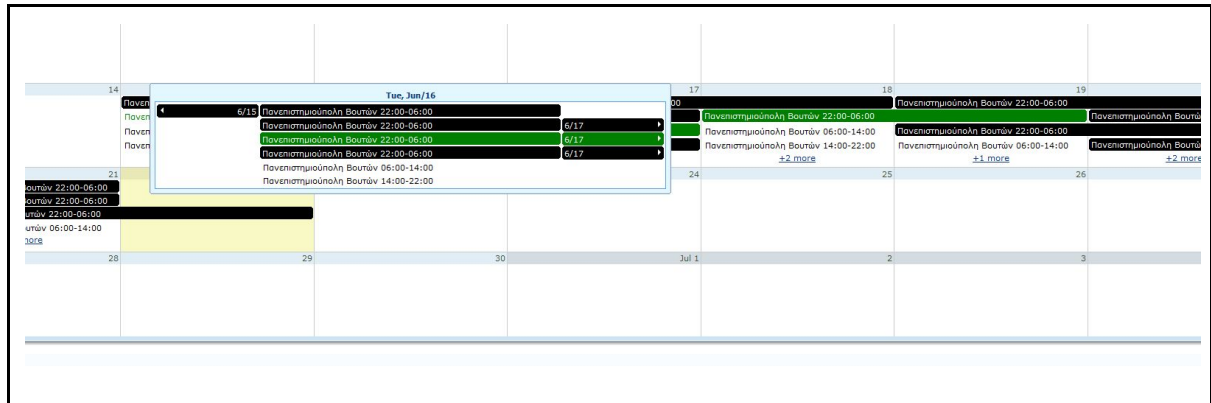
Day:



Week:



Καθώς ο admin μπορεί και βλέπει τα events όλων των εργαζομένων τα events μπορεί να μην φαίνονται όλα στην αρχική εικόνα. Υπάρχει ένα κουμπί [+X more](#) σε κάθε ημέρα που υπάρχουν πολλά events για να μπορέσει ο χρήστης να δει όλα τα events της ημέρας.



## 2.2.1. Ρομπ Σελίδα

### 2.2.1.1. Εμφάνιση πληροφοριών τρέχοντος event

Όταν ένας χρήστης κάνει κλίκ σε ένα από τα events που εμφανίζονται στο πρόγραμμα τότε εμφανίζεται ένα μικρό παράθυρο διαλόγου που επιτρέπει στον χρήστη να δει τις πληροφορίες που αφορούν το τρέχων event που επέλεξε. Οι πληροφορίες που φαίνονται είναι το ονοματεπώνυμο του εργαζομένου που αναφέρεται σε αυτό το event ως τίτλος του event. Η ημερομηνία οργάνωσης του event, η ώρα της βάρδιας και το κτήριο στο οποίο ανήκει το συγκεκριμένο event.

The screenshot displays a calendar interface with a modal dialog box open. The dialog box is titled "Event for: panos sikelis @5". It contains the following fields and options:

- Begin Date:** Jun 15, 2020
- End Date:** Jun 15, 2020
- Time Slots:** Three radio buttons are present: ☒ 06:00-14:00, ☐ 14:00-22:00, and ☐ 22:00-06:00.
- Location:** Two radio buttons are present: ☒ Πανεπιστημιούπολη Βουτών and ☐ Πανεπιστημιούπολη Κνωσού.
- Guards:** A section with a blue header labeled "Guards" and an empty list area below it.
- Buttons:** At the bottom of the dialog are three buttons: "OK", "Cancel", and "Delete".

The background calendar shows a grid for June, with dates 1, 8, 15, 22, and 29 visible. A yellow highlight is present on the 22nd. The calendar header includes "Tue", "Wed", "Thu", and "Fri" buttons, along with "Day" and "V" buttons in the top right corner.

### 2.2.1.2. Δημιουργία event

Όταν ένας χρήστης κάνει κλικ σε ένα άδειο σημείο μέσα στο κελί του ημερολογίου τότε εμφανίζεται ένα παράθυρο διαλόγου που επιτρέπει την δημιουργία ενός καινούργιου event. Μέσα από αυτό το παράθυρο ο χρήστης μπορεί να επιλέξει τον τύπο της βάρδιας, το κτίριο στο οποίο ανήκει η βάρδια και τα ονόματα των εργαζομένων για αυτό το event. Πατώντας το κουμπί OK ο χρήστης οριστικοποιεί το event. Αν ο χρήστης δεν είναι admin το event δεν αποθηκεύεται.

The screenshot shows a calendar interface with a modal dialog box titled "Event to create". The dialog box contains the following fields and options:

- Begin Date:** Jun 23, 2020
- End Date:** Jun 23, 2020
- Time Slots:** Three checkboxes for "06:00-14:00", "14:00-22:00", and "22:00-06:00".
- Locations:** Two checkboxes for "Πανεπιστημιούπολη Βουτών" and "Πανεπιστημιούπολη Κνωσού".
- Guards:** A dropdown menu with the following options:
  - ☐ 0 stratos sikelis
  - ☐ 1 panos sikelis
  - ☐ 2 kostas sikelis
  - ☐ 3 kostas3 sikelis
- Buttons:** "OK", "Cancel", and "Delete".

### 2.2.1.3. Διαγραφή επιλεγμένου event

Αν ο χρήστης κάνει κλικ σε ένα event πατώντας το κουμπί delete μπορεί να διαγράψει το επιλεγμένο event. Αν δεν είναι admin οι αλλαγές δεν αποθηκεύονται.

## 2.3. Σελίδα αίτησης και απάντησης αιτήσεων αδειών / Ιστορικό

Στην κεντρική σελίδα του ημερολογίου υπάρχει πάνω αριστερά κουμπί Request Vacation που οδηγεί στην σελίδα για την δημιουργία αίτησης για άδεια από τον χρήστη. Η σελίδα περιέχει 2 ημερολόγια για τον ορισμό του διαστήματος της άδειας. Επιπλέον στην σελίδα αυτή εμφανίζεται και το ιστορικό όλων των αδειών του χρήστη. Αν ο χρήστης είναι admin το ιστορικό εμφανίζει όλες τις άδειες που έχουν δοθεί. Επιπλέον ο admin για κάθε νέα άδεια έχει την δυνατότητα να απαντήσει αν δέχεται ή απορρίπτει την αίτηση άδειας. Η στήλη Status αναγράφει το τρέχων status της αίτησης. Αν γίνει δεκτή γράφει Accepted αλλιώς Declined. Γράφει Pending μέχρι να δοθεί απάντηση.

Admin:

Vacation Request and History Page

< Jun 2020 > < Jun 2020 >

Sun Mon Tue Wed Thu Fri Sat Sun Mon Tue Wed Thu Fri Sat

31 1 2 3 4 5 6 31 1 2 3 4 5 6

7 8 9 10 11 12 13 7 8 9 10 11 12 13

14 15 16 17 18 19 20 14 15 16 17 18 19 20

21 22 23 24 25 26 27 21 22 23 24 25 26 27

28 29 30 1 2 3 4 28 29 30 1 2 3 4

Today Today

Vacation Type:

Vacation Request

Vacation ID	Name	Surname	Starting Date	Ending Date	Type	Status	Reply
1	panos	sikelis	17.06.2020	18.06.2020	Αναρρωτική άδεια	Pending	<input type="button" value="Accept"/> <input type="button" value="Decline"/>

## Standard Employee:

Vacation Request and History Page

< Jun 2020 >

Sun Mon Tue Wed Thu Fri Sat

31 1 2 3 4 5 6

7 8 9 10 11 12 13

14 15 16 17 18 19 20

21 22 23 24 25 26 27

28 29 30 1 2 3 4

Today

< Jun 2020 >

Sun Mon Tue Wed Thu Fri Sat

31 1 2 3 4 5 6

7 8 9 10 11 12 13

14 15 16 17 18 19 20

21 22 23 24 25 26 27

28 29 30 1 2 3 4

Today

Vacation Type:

Vacation Request

Vacation ID	Name	Surname	Starting Date	Ending Date	Type	Status
1	panos	sikelis	17-06-2020	18-06-2020	Ανταποδοτική άδεια	Pending

## 2.4. Σελίδα Δημιουργίας Εβδομαδιαίου προγράμματος

Στην κύρια σελίδα υπάρχει και ένα άλλο κουμπί Create Schedule που οδηγεί στην σελίδα όπου ο χρήστης μπορεί να δημιουργήσει το τρέχων πρόγραμμα της εβδομάδας. Μόνο admin μπορεί να δει αυτό το κουμπί άρα μόνο admin μπορεί να δημιουργήσει πρόγραμμα.

Υπάρχουν 2 κτίρια για την δημιουργία προγράμματος συνεπώς οι εργαζόμενοι πρέπει να χωριστούν σε 2 ομάδες. Αυτό το κάνω με 2 τρόπους. Σε αυτή τη σελίδα μπορεί ο χρήστης να διαλέξει είτε το σύστημα να δημιουργήσει αυτές τις ομάδες ή ο χρήστης μπορεί να αναθέσει έναν έναν τους εργαζομένους στο κτίριο που θέλει εκείνος. Εάν επιλέξει ο χρήστης να αναθέσει τους εργαζομένους στα κτίρια θα πρέπει να τους αναθέσει όλους. Αν κάποιος δεν έχει ανατεθεί δεν το σύστημα εμφανίζει μήνυμα λάθους.

Schedule Creator

Description:  
Select the employees to be assigned on each building for the schedule to be created.  
Please note that if ANY employee is not assigned a building,  
your changes will be IGNORED and the system will auto-assign the employees.

Auto-generate Employee Building Assignments

User ID	Name	Surname	Chosen Building
0	stratos	sikelis	
1	panos	sikelis	
2	kostas	sikelis	
3	kostas3	sikelis	
4	kostas4	sikelis	
5	kostas5	sikelis	
6	kostas6	sikelis	
7	kostas7	sikelis	
8	kostas8	sikelis	
9	kostas9	sikelis	
10	kostas10	sikelis	
11	kostas11	sikelis	
12	kostas12	sikelis	

Submit

## 2.5. Σελίδα ελέγχου μηνιαίου συνόλου βαρδιών

Πατώντας το κουμπί Billing στην κύρια σελίδα ο χρήστης οδηγείται σε μία σελίδα με ένα πίνακα όπου αναγράφονται όλοι οι εργαζόμενοι φύλακες και οι συνολικές βάρδιες ανά τύπο που έχουν ολοκληρώσει τον τρέχων μήνα. Οι βάρδιες χωρίζονται σε κανονικές, βραδινές, αργίες και βραδινές αργίες.

Employee Billings						
Monthly History						
User ID	Name	Surname	Normal Shifts	Night Shifts	Holiday Shifts	Night-Holiday Shifts
0	stratos	sikelis	0	0	0	0
1	panos	sikelis	0	0	0	0
2	kostas	sikelis	0	0	0	0
3	kostas3	sikelis	0	0	0	0
4	kostas4	sikelis	0	0	0	0
5	kostas5	sikelis	0	0	0	0
6	kostas6	sikelis	0	0	0	0
7	kostas7	sikelis	0	0	0	0
8	kostas8	sikelis	0	0	0	0
9	kostas9	sikelis	0	0	0	0
10	kostas10	sikelis	0	0	0	0
11	kostas11	sikelis	0	0	0	0
12	kostas12	sikelis	0	0	0	0

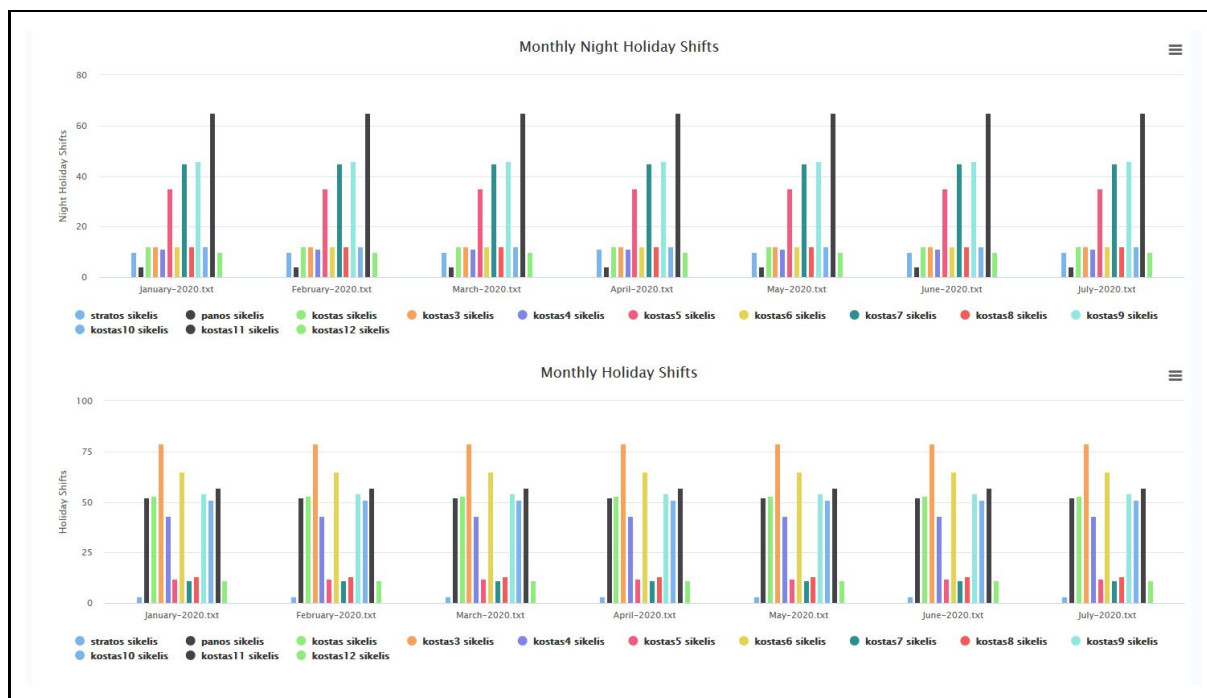


### 2.5.1. Σελίδα ελέγχου μηνιαίου συνόλου βαρδιών

Το κουμπί Monthly History στη σελίδα Billing οδηγεί σε μια σελίδα που περιέχει έναν πίνακα που επιτρέπει στον χρήστη να ελέγξει τις συνολικές βάρδιες κάθε φύλακα για τον επιλεγμένο μήνα. Υπάρχουν filters για να διαλέξει ο χρήστης τον μήνα.

History						
Select Month: <span>January</span>			Select Year: <span>2020</span>		<span>Submit</span>	
User ID	Name	Surname	Normal Shifts	Night Shifts	Holiday Shifts	Night-Holiday Shifts
0	stratos	sikelis	0	0	3	10
1	panos	sikelis	0	0	52	4
2	kostas	sikelis	0	0	53	12
3	kostas3	sikelis	0	0	79	12
4	kostas4	sikelis	0	0	43	11
5	kostas5	sikelis	0	0	12	35
6	kostas6	sikelis	0	0	65	12
7	kostas7	sikelis	0	0	11	45
8	kostas8	sikelis	0	0	13	12
9	kostas9	sikelis	0	0	54	46
10	kostas10	sikelis	0	0	51	12
11	kostas11	sikelis	0	0	57	65
12	kostas12	sikelis	0	0	11	10

Επιπλέον η σελίδα περιέχει γραφήματα που δείχνουν τις συνολικές μηνιαίες βάρδιες για τις αργίες κάθε φύλακα χωρισμένες σε κανονικές και βραδινές. Τα γραφήματα δείχνουν όλους τους αποθηκευμένους μήνες και όλους τους εργαζόμενους.



### 3. Εργαλεία Κατασκευής Εφαρμογής / Συστήματος

Για την κατασκευή της εφαρμογής χρησιμοποιήθηκαν τα παρακάτω εργαλεία.

- **Java SE Development Kit** version 8.
- **Eclipse IDE**.
- **ZK Studio**.
- **Maven**.
- **MySQL 8.0 CE**. Χρησιμοποιήθηκε το **MySQL Workbench** app για την κατασκευή πινάκων και έλεγχο δεδομένων.
- **Hibernate ORM** για την επικοινωνία του webapp με την βάση δεδομένων.
- **Jetty Server**.
- **Quartz Scheduler** για τον χρονοπρογραμματισμό και την αυτόματη εκτέλεση μεθόδων.
- **Optaplaner** για την δημιουργία του χρονοπρογραμματισμού των φυλάκων.  
Είναι εργαλείο για την επίλυση περιορισμών (Open Source Constraint Solver).

#### 3.1. ZK Studio - ZK framework

Το ZK framework είναι ένα εργαλείο το οποίο επιτρέπει την δημιουργία διεπαφών χρήστη με γραφικά για διαδικτυακές εφαρμογές. Είναι open-source βασισμένο στο Ajax Web application framework. Το ZK χρησιμοποιεί server-centric προσέγγιση για τον συγχρονισμό των component της σελίδας και του event pipe-lining μεταξύ των clients και των server.

### **3.2. Maven**

Το Maven είναι ένα εργαλείο αυτοματοποίησης της διαδικασίας δημιουργίας μιας εφαρμογής. Επιτρέπει την αυτόματη προσθήκη βιβλιοθηκών και άλλων στοιχείων που χρειάζεται η εφαρμογή κατά την λειτουργία της. Ο χρήστης χρειάζεται να ορίσει ποιές βιβλιοθήκες χρειάζεται για το project του και το Maven βρήσκει και εισάγει όλα τα απαραίτητα αρχεία.

### **3.3. MySQL**

Το MySQL είναι ένα open-source σύστημα σχεσιακής διαχείρισης βάσεων δεδομένων. Το σύστημα επιτρέπει την δημιουργία πινάκων όπου τα στοιχεία τους έχουν σχέσεις μεταξύ τους για την γρήγορη και αποτελεσματική αποθήκευση και αναζήτηση δεδομένων. Χρησιμοποίησα το MySQL Workbench για την εύκολη χρήση και για τον έλεγχο της βάσης. Το Workbench είναι μία εφαρμογή που επιτρέπει μέσω ενός GUI την εύκολη διαχείριση της βάσης. Δημιουργία πινάκων, έλεγχο δεδομένων.

### **3.4. Hibernate ORM**

Το Hibernate ORM είναι ένα εργαλείο αντιστοίχισης αντικειμένων με σχέσεις βάσεων δεδομένων για Java εφαρμογές. Επιτρέπει την αντιστοίχιση αντικειμένων Java με πίνακες στην βάση δεδομένων για την εύκολη, γρήγορη και ασφαλή επικοινωνία της Java εφαρμογής με την βάση δεδομένων.

### **3.5. Quartz Scheduler**

Το Quartz είναι μία βιβλιοθήκη για τον χρονοπρογραμματισμό μεθόδων σε Java εφαρμογές. Το Quartz χρησιμοποιείται για εφαρμογές εταιρικής κλάσης για την υποστήριξη της ροής εργασιών της διαδικασίας, των ενεργειών διαχείρισης συστήματος και την παροχή έγκαιρων υπηρεσιών εντός των εφαρμογών.

### **3.6. Jetty Server**

Ο Jetty Server είναι ένας Java HTTP web server και Java Servlet container. Επιτρέπει την λειτουργία μιας Java web εφαρμογής.

### **3.7. OptaPlanner**

Το Optaplanner είναι ένα open source εργαλείο γραμμένο σε Java για την επίλυση προβλημάτων με περιορισμούς με τη κατασκευή heuristic και metaheuristic αλγορίθμους.

## 4. Backend Αρχιτεκτονική Συστήματος.

Στις παρακάτω ενότητες θα αναλυθεί κάθε feature και θα εξηγηθούν όλες οι συναφείς κλάσεις, πίνακες και ζυγ αρχεία για αυτό το feature.

### Database Tables:

- **login.** Κρατάει για κάθε εργαζόμενο το username (String), password (String), id (Integer), admin (binary bit)
- **user.** Κρατάει για κάθε εργαζόμενο το id (Integer), name (String), surname (String), standardShift (Integer), nightShift (Integer), holidayNightShift (Integer), holidayShift (Integer).
- **calendarevent.** Κρατάει για κάθε event το id του event (Integer), day (Integer), month (Integer), year (Integer), shift (Integer), idUser (Integer), building (Integer).
- **vacationdays.** Κρατάει για κάθε αίτηση άδειας το id (Integer), user\_id (Integer), starting\_date (String), ending\_date (String), type (String), status (Integer), week\_num\_start (Integer), week\_num\_end (Integer)
- **scheduled\_week.** Κρατάει τον αριθμό της τελευταίας εβδομάδας που εκδόθηκε πρόγραμμα, id (Integer), scheduled\_week (Integer).

## 4.1. Σύστημα Login.

### Java classes:

- login.java. Java Entity για την επικοινωνία webapp και login table.
- LoginController.java. Controller class για την λειτουργία του login.

### ZUL files:

- login.zul. Welcome σελίδα της εφαρμογής.

Το αρχείο login.zul περιέχει 2 textbox τα οποία γίνονται Wire στο LoginController.java. Στον controller όταν πατηθεί το κουμπί login εκτελείται μία μέθοδος που εξάγει τα δεδομένα από τα wired textboxes και φτιάχνει ένα select query για τον πίνακα login στη βάση δεδομένων. Αν το select query επιστρέψει αποτελέσματα για το ζευγάρι username-password τότε καλείται μία redirect μέθοδος που ελέγχει αν ο χρήστης που κάνει login είναι admin ή όχι. Αν δεν επιστρέψει αποτέλεσμα το query τότε εμφανίζεται μήνυμα λάθους. Αν είναι admin γίνεται redirect στην κύρια σελίδα calendar για τους admin, αλλιώς γίνεται redirect στην απλή κύρια σελίδα calendar. Επιπλέον κάνω url encoding για να περάσω 2 παραμέτρους, το idUser και τον αν είναι admin ή όχι (binary 0,1).

## 4.2. Σύστημα Calendar

### Java classes:

- calendarController.java. Κύρια κλάση για την λειτουργία της σελίδας.
- calendarevent.java. Entity κλάση για την επικοινωνία webapp και calendarevent table.
- user.java.Entity κλάση για την επικοινωνία webapp και user table.
- CalendarPopup.java. Κύρια κλάση για το popup του calendar.

### ZUL files:

- calendar.zul. Κύρια σελίδα όταν κάνει login ένας admin.
- calendar\_non\_admin.zul. Κύρια σελίδα όταν κάνει login ένας απλός χρήστης.
- calendar\_popup.zul

### 4.2.1. Παρουσίαση των Event

Όταν γίνει redirect σε calendar σελίδα, το πρώτο πράγμα που γίνεται είναι να πάρω από το URL το userID και το αν είναι admin. Μετά σύμφωνα με αυτά τα στοιχεία κάνω query select \* από τον πίνακα calendarevent για να πάρω όλα τα events που αφορούν τον χρήστη που έκανα login. Αν είναι admin φέρνω όλο το table, αν όχι φέρνω μόνο τα event που αντιστοιχούν στο userID. Ταυτόχρονα κάνω και ένα δεύτερο query select iduser,name,surname στον πίνακα user για να μπορώ να εμφανίσω το ονοματεπώνυμο του χρήστη που θέλω. Χρειάζομαι το ονοματεπώνυμο για το popup.

Για να εμφανίσει το calendar κάθε event πρέπει πρώτα να δημιουργήσω ένα αντικείμενο τύπου SimpleCalendarEvent. Σε αυτό το αντικείμενο ορίζω το Content και το Title ανάλογα με τα δεδομένα που πείρα ως απάντηση από τα queries. Το Content περιέχει το κτίριο και το πότε αρχίζει και πότε τελειώνει η βάρδια. Το Title περιέχει το ονοματεπώνυμο του φύλακα και το id του event. Εισάγω αυτό το event στο μοντέλο του calendar (SimpleCalendarModel). Μετά ορίζω το μοντέλο του calendar να είναι το μοντέλο όπου έβαλα όλα τα events.

#### 4.2.2. Λειτουργία του Calendar Popup

Το ZK framework επιτρέπει τον χωρισμό της δημιουργίας και της επεξεργασίας των event στο calendar μέσω 2 ξεχωριστών συναρτήσεων.

##### 1. **public void onEventCreate\$calendar(CalendarEvent event)**

Αυτή η μέθοδος επιτρέπει την δημιουργία ενός καινούργιου event. Η συνάρτηση εκτελείται όταν ο χρήστης κάνει κλίκ σε κενό σημείο σε κελί του calendar. Η μέθοδος αρχικά αλλάζει το visibility του popup σε "true". Καθώς το popup περιέχει 2 dateboxes πρέπει αλλάξω την ημερομηνία που δείχνουν, διότι δείχνουν την τρέχουσα ημέρα by default. Παίρνω από το event το BeginDate και EndDate και μετά ορίζω τα αντίστοιχα dateboxes. Στην κλάση CalendarPopup υπάρχουν global μεταβλητές που ορίζονται από τις αντίστοιχες συναρτήσεις τους για την επιλογή της βάρδιας, του κτιρίου και του εργαζομένου. Οι επιλογές γίνονται μέσω **onCheck(CheckEvent event)** μεθόδους. Για το OK button υπάρχει αντίστοιχη συνάρτηση που παίρνει τις global μεταβλητές, παίρνει το max id (**select max(idCalendarEvent)** ) από τα calendar events που είναι αποθηκευμένα στη βάση, το μεγαλώνει κατά ένα και μετά με ένα query **insert into calendarevent** αποθηκεύω το καινούργιο event.

Όταν κάνει login ένας admin και μπει στην κύρια σελίδα εκτελείται ένα query select iduser, name, surname και τα εισάγω σε μία λίστα την οποία δίνω στο popup για να μπορέσω να γεμίσω το Listbox του popup για την ανάθεση των εργαζομένων στο event. Από τις επιλογές του χρήστη στο Listbox μπορώ να πάρω το iduser και για κάθε επιλεγμένο φύλακα να φτιάξω ένα καινούργιο event για εκείνον.



## 2. `public void onEventEdit$calendar(CalendarEvent event)`

Αυτή η μέθοδος εκτελείται όταν ο χρήστης επιλέξει κάποιο event απο το calendar. Από το argument event μπορώ να πάρω από το Title το **id** του και από το Content το κτίριο και την βάρδια. Σύμφωνα με αυτά τα στοιχεία ορίζω τα κατάλληλα Checkbox του popup για να αντιπροσωπευουν τα δεδομένα του event. Πατώντας OK ή Cancel δεν αλλάζει κάτι στο event. Η μόνη αλλαγή που μπορεί να γίνει είναι όταν ο χρήστης πατήσει Delete όπου καλείται μια συνάρτηση που ακούει σε αυτό το button και μέσω της global μεταβλητής **eventID** που την γεμίζει η **onEventEdit** η delete μέθοδος εκτελεί ένα query delete με αυτό το eventID.

Επιπλέον έχω φτιάξει μία συνάρτηση που μεταφράζει την επιστρεφόμενη τιμή του event για την StartDate και EndDate από όνομα μήνα ολογράφως (Jan,Feb...) σε αριθμό (01,02...) για τον ορισμό των dateboxes.

Οι **onCheck** για τα checkboxes που υπάρχουν στο popup δεν γεμίζουν μόνο τις global μεταβλητές για το eventCreate αλλά αλλάζουν για τις τιμές ( **setChecked** ) των άλλων επιλογών για να υπάρχει μόνο ένα checkbox επιλεγμένο από την ίδια ομάδα. Για παράδειγμα αν είναι επιλεγμένο από πριν η Πανεπιστημιούπολη Βουτών και μετά επιλεγεί η Πανεπιστημιούπολη Κνωσού τότε πρέπει η Πανεπιστημιούπολη Βουτών να γίνει unchecked και checked η Πανεπιστημιούπολη Κνωσού.

### 4.3. Σύστημα Request Vacation and Vacation History

#### Java classes:

- VacationController.java. Κύρια κλάση για την λειτουργία της σελίδας.
- vacationdays.java. Entity κλάση για την επικοινωνία webapp και vacationdays table.
- VacationList.java. Κλάση για το model του Listbox με τις αιτήσεις αδειών.
- WeekNumber.java. Κλάση για την εύρεση αριθμού εβδομάδας μέσα από ημερομηνία.

#### ZUL files:

- vacation.zul. Κύρια σελίδα όταν κάνει login ένας admin για έλεγχο για απάντηση σε αιτήσεις αδειών.
- vacation\_non\_admin.zul. Κύρια σελίδα όταν κάνει login ένας απλός χρήστης για την δημιουργία αίτησης άδειας και έλεγχο ιστορικού.

#### 4.3.1 Ορισμός Listbox και απάντηση αιτήσεων αδειών

Η σελίδα περιέχει 2 dateboxes και ένα textbox που έγιναν Wired στον VacationController. Από αυτά τα components μπορώ να εξάγω την πρώτη και τελευταία ημέρα της άδειας καθώς και τον τον τύπο της. Από τον διαχωρισμό admin και κανονικού χρήστη κάνω ένα query select \* από τον πίνακα vacationdays, ανάλογα με το id του χρήστη, και εμφανίζω στο listbox το ιστορικό των αδειών. Αν δεν είναι admin το query εκτελείται μόνο για το id του χρήστη. Για κάθε αποτέλεσμα του query δημιουργώ ένα αντικείμενο VacationList και εισάγω τα δεδομένα της γραμμής του πίνακα. Μετά αυτό το αντικείμενο το εισάγω στο ListModelList του Listbox για να το περάσω στο frontend και να γεμίσω το Listbox. Στον VacationController έχω ένα constructor που φτιάχνει ένα ListModelList object για το Listbox και υπάρχει και ένας getter για να επιστρέφει ο controller το model.

Στο vacation.zul σε κάθε γραμμή του listbox που γεμίζει με τα στοιχεία της αίτησης στην τελευταία στήλη "Reply" δημιουργούνται 2 buttons για την αποδοχή ή απόρριψή της αίτησης.

Αυτά τα κουμπιά δημιουργούν δικό τους id στο zul σύμφωνα με τον τύπο του κουμπιού και το id της αίτησης. Δηλαδή το accept button για την άδεια με id = 10 θα έχει id = Accept+10. Καθώς αυτά τα κουμπιά είναι δυναμικά σύμφωνα με το listbox δεν μπορούν να γίνουν Wired από το id τους. Για να μπορέσω να βρώ πιο button

πατήθηκε και σε ποια άδεια ανήκει πρέπει να πάρω ένα **MouseEvent event**. Από αυτό το event μπορώ να κάνω **getTarget()** και να βρω ποιο κουμπί πατήθηκε. Μετά μπορώ να πάρω το id του button με **getId()** και έτσι θα ξέρω τον τύπο του κουμπιού για την απάντηση και το id της αίτησης της άδειας. Με το id μπορώ να ψάξω μετά το ListModelList του Listbox και να αλλάξω το Status argument από Pending σε Accept ή Rejected, και κάνω update το ListModelList. Ταυτόχρονα κάνω ενα **session.get(vacationdays.class, vid)** όπου βρίσκω την αίτηση στον βάση και μετά κάνω σεντ το Status argument και μετά κάνω **commit** και **session.close()**.

#### 4.3.2 Κατοχύρωση αίτησης άδειας

Όταν πατηθεί το κουμπί submit εκτελείται μία συνάρτηση για την αποθήκευση της αίτησης άδειας. Αρχικά παίρνω από τα 2 dateboxes τις ημερομηνίες και τις συγκρίνω έτσι ώστε η αρχή της άδειας να είναι πριν το τέλος της. Αφού έχω δεκτές ημερομηνίες για αρχή και τέλος τότε βρίσκω αυτές οι 2 ημερομηνίες σε ποιές εβδομάδες ανήκουν μέσω της **getWeekNumber(String date)** στη **WeekNumber.java class**. Μετά δημιουργώ ένα **vacationdays** object και εισάγω σε αυτό τις τιμές, starting date, ending date, type ( textbox wire) , user id ( από το login URL parameter ), status = 0 (Pending), week start, week end. Χρειάζεται να γνωρίζω τις εβδομάδες για την δημιουργία του προγράμματος αργότερα. Μετά εκτελώ ένα **query select max(vacation\_id)** για να το μεγαλώσω και να το βάλω στην καινούργια αίτηση. Μετά από το userID κάνω ένα **session.get(user.class, userID)** για να πάρω το ονοματεπώνυμο για να το εισάγω στο ListModelList του Listbox μαζί με τα προηγούμενα δεδομένα που μου έδωσε στην αίτηση.

#### 4.4. Σύστημα Ιστορικού Βαρδιών

Αυτό το σύστημα χωρίζεται σε 2 κομμάτια.

1. Αποθήκευση βαρδιών ημέρας και εμφάνιση συνολικού αριθμού βαρδιών του κάθε φύλακα.
2. Εμφάνιση ιστορικού συνολικών βαρδιών παλαιότερων μηνών και εμφάνιση γραφημάτων.

##### 4.4.1. Αποθήκευση βαρδιών ημέρας και εμφάνιση συνολικού αριθμού βαρδιών του κάθε φύλακα

###### Java classes:

- `billingController.java`. Κύρια κλάση για εμφάνιση των συνολικών ολοκληρωμένων βαρδιών κάθε εργαζόμενου.
- `ShiftCompletion.java`. Ολοκλήρωση και αποθήκευση βαρδιών ημέρα στους εργαζόμενους.
- `UserList.java`. Βοηθητικό object για την εισαγωγή δεδομένων στο Listbox.

###### ZUL files:

- `billing.zul`. Κύρια σελίδα για την εμφάνιση των συνολικών βαρδιών του μήνα.

Ο `billingController` είναι υπεύθυνος για να εισάγει τα δεδομένα του table `user` μέσα στο Listbox της σελίδας. Ο controller εκτελεί ένα **query select \* from user** και για κάθε γραμμή του πίνακα παίρνει τα δεδομένα και τα εισάγει στο `ListModelList` του Listbox. Για κάθε γραμμή δημιουργείται ένα αντικείμενο τύπου `UserList.java` που αποθηκεύει τα δεδομένα που πρέπει να εμφανιστούν. Μετά αυτό το αντικείμενο εισάγεται στο Model του Listbox.

Η κλάση `billingController.java` έχει constructor για την δημιουργία του **ListModelList<UserList>()**. Υπάρχει και getter για το model.

Η κλάση **ShiftCompletion.java** περιέχει μία μέθοδο που είναι χρονοπρογραμματίσιμη από το εργαλείο Quartz. Ο σκοπός της κλάσης είναι κάθε βράδυ να “διαβάζει” όλα τα event (βάρδιες) της ημέρας και τις προσθέτει στις συμπληρωμένες βάρδιες του αντίστοιχου χρήστη στον πίνακα user της βάσης.

Η μέθοδος ελέγχει αν η τρέχουσα ημέρα είναι καθημερινή ή αργία. Οι σταθερές αργίες έχουν απλώς οριστεί ως static μεταβλητές. Οι αργίες που αλλάζουν ημερομηνία σύμφωνα με την ημερομηνία του πάσχα υπολογίζονται αφού πρώτα υπολογιστεί η ημέρα του Πάσχα το τρέχων έτος. Για τον υπολογισμό του Πάσχα χρησιμοποιώ τον **Meeus's Julian** αλγόριθμο. Σύμφωνα με την ημερομηνία που μου επιστρέφει ο αλγόριθμος μπορώ να υπολογίσω τις υπόλοιπες κινητές γιορτές. Εισάγω όλες τις αργίες σε ένα σύνολο για τις συγκρίνω όλες με την τρέχουσα ημέρα για να δω αν είναι αργία ή όχι.

Αρχικά εκτελώ ένα **query select** στον πίνακα **calendarevent** με παραμέτρους την τρέχουσα ημερομηνία χωρισμένη σε ημέρα-μήνα-έτος για να πάρω όλα τα **event** της ημέρας. Γνωρίζοντας αν είναι αργία η ημερομηνία και γνωρίζοντας τον τύπο της βάρδιας από την απάντηση του πίνακα με τα events μπορώ να καθορίσω ποια μεταβλητή του πίνακα **user** πρέπει να αυξήσω. Εκτελώ ένα **query session.get(user.class, idUser)** όπου το idUser το παίρνω από το **event** και μετά εκτελώ τον αντίστοιχο setter της μεταβλητής που πρέπει να αυξήσω.

#### 4.4.2. Εμφάνιση ιστορικού συνολικών βαρδιών παλαιότερων μηνών και εμφάνιση γραφημάτων

##### Java classes:

- historyController.java. Κύρια κλάση για εμφάνιση ιστορικού ολοκληρωμένων βαρδιών.
- StoreMonthlyShifts.java. Δημιουργία ιστορικού στο τέλος του μήνα.
- UserList.java. Βοηθητικό object για την εισαγωγή δεδομένων στο Listbox.
- Year.java Βοηθητικό object για την εισαγωγή του έτους στο ιστορικό
- Months.java Βοηθητικό Object για την εισαγωγή μήνα στο ιστορικό

##### ZUL files:

- history.zul. Κύρια σελίδα για την εμφάνιση ιστορικού.

Η συνάρτηση **redirect()** στον **billingController** ακούει στο κουμπί **Monthly History** για να γίνει **redirect** στη σελίδα ιστορικού.

Ο **historyController** έχει 2 συναρτήσεις **changeMonth** και **changeYear** που ακούνε στα αντιστοιχα **selectboxes** έτσι ώστε να μπορεί ο χρήστης να επιλέξει για ποιόν μήνα θέλει να δει το ιστορικό. Για αυτά τα 2 **selectboxes** υπάρχουν **ListModelList<Months>** και **ListModelList<Year>** για τον ορισμό των πιθανών επιλογών αντίστοιχα για μήνες και έτη. Ο controller έχει 2 global μεταβλητές που αποθηκεύουν την τρέχουσα τιμή των 2 **selectboxes**, μία για το μήνα και μια για το έτος. Όταν πατηθεί το κουμπί **submit** ο listener στον controller εκτελεί μία μέθοδο όπου ψάχνει το κατάλληλο αρχείο για να το διαβάσει σύμφωνα με το όνομα του αρχείου και τις επιλογές του χρήστη. Η μέθοδος βρίσκει, διαβάζει το αρχείο ανα γραμμή και τα εισάγει **ListModelList<UserList>** τα δεδομένα προς εμφάνιση. Αν δεν βρεθεί αρχείο τότε εμφανίζεται μήνυμα λάθους.

Επιπλέον στην σελίδα υπάρχουν 2 ραβδογράμματα που δείχνουν τις βάρδιες για τις αργίες όλων των μηνών που υπάρχει ιστορικό. Το πρώτο δείχνει τις νυχτερινές βάρδιες σε αργίες. Το δεύτερο δείχνει τις βάρδιες σε αργίες πρωί και μεσημέρι. Ο controller διαβάζει ένα ένα τα αρχεία, περνάει τα ονόματα από μία μέθοδο **order** έτσι ώστε οι μήνες να είναι στην σωστή σειρά και έπειτα τα αρχεία διαβάζονται γραμμή γραμμή. Το κάθε ραβδόγραμμα έχει το δικό του **Model** τύπου **DefaultCategoryModel** στο οποίο εισάγω το όνομα του αρχείου και έναν έναν τους εργαζόμενους που αναγράφονται στο αρχείο του ιστορικού και το δεδομένο που αντιστοιχεί στο κατάλληλο Model.

Η **StoreMonthlyShift.java** κλάση έχει μία μέθοδο που είναι υπεύθυνη για την αποθήκευση των συνολικών μηνιαίων βαρδιών κάθε φύλακα που περιέχονται στο πίνακα **user**. Η μέθοδος ονομάζεται **execute(JobExecutionContext context)** και χρονοπρογραμματίζεται από το εργαλείο **Quartz**. Η μέθοδος αρχικά εκτελεί ένα **query select** στον πίνακα **user**. Μετά βρήσκει τον μήνα και το έτος μέσω **LocalDate** και συντάσσει ένα String για το όνομα του αρχείου της μορφής **month-year.txt** όπου θα αποθηκευτούν όλα τα δεδομένα που πήρα από τον πίνακα της βάσης. Τα δεδομένα τα γράφω στο αρχείο μέσω ενός **FileWriter(filename)** όπου το filename είναι το αρχείο που μόλις έφτιαξα. Για κάθε χρήστη που πήρα ως απάντηση στο προηγούμενο query καθώς γράφω τα δεδομένα στο αρχείο παράλληλα εκτελώ ένα **query update** έτσι ώστε να μηδενίσω τον πίνακα της βάσης για να είναι έτοιμος για τον επόμενο μήνα. Η μέθοδος αυτή πρέπει να χρονοπρογραμματιστεί να εκτελεστεί μετά την εκτέλεση της **ShiftCompletion** την τελευταία μέρα του μήνα.

#### 4.5. Σύστημα Δημιουργίας προγράμματος φυλάκων

##### Java classes:

- `schedule_creator_Controller.java`. Κύρια κλάση για τον χωρισμό των φυλάκων στα 2 κτίρια.
- `Possible_Worker_set.java`. Δημιουργία πιθανών συνόλων εργαζομένων.
- `UserList.java`. Βοηθητικό object για την εισαγωγή δεδομένων στο Listbox.
- `vacationdays.java`. Βοηθητικό object για την επικοινωνία webapp με πίνακα βάσης `vacationdays`.
- `UserVacDays.java`. Βοηθητικό object για την αποθήκευση των ημερών άδειας της τρέχουσας ημέρας.
- `TotalUserVacDays.java`. Βοηθητικό object για την αποθήκευση των συνολικών ημερών άδειας μέσα στην εβδομάδα σε περίπτωση που ο χρήστης έχει πολλαπλές άδειες στην ίδια εβδομάδα.
- `employeeMapping.java`. Βοηθητικό object για την αντιστοίχιση `user_id` με `index` πίνακα.
- `ScheduleGenerator.java`. Κύρια κλάση για την έκδοση του εβδομαδιαίου προγράμματος.
- `ShiftSchedule.java`. Planning Solution για το Optaplanner εργαλείο.
  - Αυτό το αντικείμενο κρατάει μία λίστα με όλα τα `uid` των φυλάκων που θα μπουν στο πρόγραμμα
  - Κρατάει μία λίστα τύπου **ShiftAssignment.java** που κρατάει ένα ζευγάρι 1 integer εργαζομένου `id` και 1 αντικείμενο τύπου **Shift.java** που κρατάει 1 integer για τον αριθμό βάρδιας.
- `ScoreCalculator.java`. Κλάση βαθμολογία πιθανών λύσεων για την έκδοση προγράμματος.
- `finalShift.java`. Εκτικείμενο εργαζομένου για την λίστα προγράμματος.

##### ZUL files:

- `schedule_creator.zul`. Κύρια σελίδα για την επιλογή του τρόπου δημιουργίας του προγράμματος των φυλάκων.

Αρχικά όταν φορτώσει η σελίδα εκτελείται μία μέθοδος που γεμίζει το **ListModel** του **ListBox** με τα **UserList** object που περιέχουν τα ονόματα των εργαζομένων από τον πίνακα **user** της βάσης. Εμφανίζω στο ListBox το `user id`, το όνομα και το επώνυμο.

Όταν ένας admin χρήστης εισέλθει στην σελίδα αυτή του εμφανίζονται 2 επιλογές

1. Η πρώτη επιλογή είναι να αφήσει το σύστημα να χωρίσει τους φύλακες στα 2 κτίρια προς χρονοπρογραμματισμό.
2. Η δεύτερη επιλογή ζητάει από τον χρήστη να ορίσει κάθε εργαζόμενο σε ποιο κτίριο ανήκει για αυτή τη βδομάδα.

#### 4.5.1. Αυτόματος χωρισμός φυλάκων

Ο χωρισμός των φυλάκων γίνεται σύμφωνα με τις ημέρες άδειας που έχει ο καθένας για εκείνη την εβδομάδα. Ουσιαστικά το σύστημα προσπαθεί οι συνολικές βάρδιες που μπορούν να ολοκληρωθούν στα 2 κτίρια να έχουν την ελάχιστη δυνατή διαφορά. Αυτό γίνεται βρίσκοντας πρώτα την ελάχιστη δυνατή διαφορά 2 αθροισμάτων. Αυτά τα 2 αθροίσματα είναι οι συνολικές ημέρες άδειας που έχουν οι εργαζόμενοι την συγκεκριμένη εβδομάδα.

Αρχικά όταν ο χρήστης πατήσει το κουμπί **Auto-generate Employee Building Assignments** εκτελείται μία μέθοδος ( **auto\_assignments()** ) που ακούει στο κουμπί αυτό. Πρώτα εκτελώ ένα **session.get(scheduled\_week.class, 0)** για να πάρω από τον πίνακα **scheduled\_week** τον αριθμό της τελευταίας εβδομάδας όπου δημιουργήθηκε πρόγραμμα. Αυτόν τον αριθμό τον αυξάνω κατά ένα και μετά τον αποθηκεύω πάλι στον πίνακα.

Μετά την αποθήκευση της εβδομάδας εκτελείται η συνάρτηση **roster\_generator()**. Αυτή η συνάρτηση βρήσκει πρώτα βρήσκει πρώτα ποιοί εργαζόμενοι έχουν άδειες την τρέχουσα εβδομάδα, πόσες ημέρες και ποιές ημέρες έχουν άδεια. Για να βρεθεί αυτό εκτελείται η συνάρτηση **totalVacDays()** η οποία αρχικά βρήσκει ποιές είναι η ημερομηνίες της εβδομάδας όπου θέλω να βγάλω το πρόγραμμα. Η συνάρτηση που βρήσκει τις ημερομηνίες την εβδομάδας ονομάζεται **getCurrentWeek()**. Αυτή η συνάρτηση παίρνει από τον πίνακα **scheduled\_week** της βάσης τον αριθμό της εβδομάδας και επιστρέφει έναν πίνακα **String[]** μήκους 7 με κάθε ημερομηνία της εβδομάδος π.χ. 23-04-2020, 24-04-2020..... . Επιπλέον έχει οριστεί ότι η πρώτη ημέρα της εβδομάδας είναι η Δευτέρα.

Όταν επιστρέψω από την **getCurrentWeek()** στην **totalVacDays()** εκτελώ ένα **query select** στον πίνακα **vacationdays** με παραμέτρους **status = 1** , **week\_num\_start <= current\_week** και **week\_num\_end >= current\_week**. Με αυτό το **query** παίρνω όλους του εργαζόμενους με άδειες που έχουν overlap με την



τρέχουσα εβδομάδα. Μετά κατασκευάζω μία λίστα από **vacationdays** αντικείμενα που περιέχουν τα αποτελέσματα του **query**. Αυτή την λίστα μαζί με τον πίνακα ημερών εβδομάδος τα περνάει στη συνάρτηση **getVacDays(String[] currWeek, List<vacationdays> vacDays)** για να βρω ποιές ημέρες της εβδομάδος οι φύλακες έχουν άδεια. Για κάθε αντικείμενο της λίστας **vacDays** ορίζω 2 μεταβλητές **start, end** που δείχνουν την αρχική ημέρα και την τελική ημέρα της άδειας του χρήστη. Μέσο της **start** και **end** δημιουργώ μία λίστα **betweenDays** με όλες τις ημερομηνίες ανάμεσα στην αρχή και το τέλος της άδειας. Έτσι έχω όλες τις ημερομηνίες άδειας ενός εργαζομένου. Μετά βρήσκω τις κοινές ημερομηνίες μεταξύ τρέχουσας εβδομάδας **currWeek** και λίστα ημερών άδειας **betweenDays**. Κάθε κοινό στοιχείο τους το εισάγω σε μία άλλη λίστα με όνομα **meres** τύπου **String**. Κατασκευάζω μία λίστα τύπου **UserVacDays** σε αυτή τη λίστα αποθηκεύω το **id** του φύλακα για το οποία βρήκα τις ημέρες άδειας της εβδομάδας και αποθηκεύω επίσης την λίστα **meres**. Αυτή την **UserVacDays** λίστα την επιστρέφω στην **totalVacDays** μέθοδο.

Μέσα στην κλάση **schedule\_creator\_Controller** έχει οριστεί μία global λίστα τύπου **TotalUserVacDays** που κρατάει το **sum** όλων των ημερών άδειας της εβδομάδας για του φύλακα. Επιπλέον ετοιμάζει και ένα **id** ανεξάρτητο του **user\_id** για mapping φύλακα **user\_id** με **id** όπου το κάθε **id** ανήκει στο εύρος [0 - πλήθος φυλάκων) ενώ το **user\_id** είναι ανεξάρτητο από όλα. Θα το χρειαστώ το mapping για στην δημιουργία του χρονοπρογράμματος.

Όταν ολοκληρωθεί ο ορισμός της global λίστας **tuvd** επιστρέφω στη **roster\_generator()**. Τώρα αφού έχω το σύνολο των ημερών άδειας για κάθε φύλακα μπορώ να χωρίσω τους εργαζόμενους σε 2 σύνολα όπου η διαφορά τους είναι η ελάχιστη. Για να βρώ αυτή την ελάχιστη διαφορά καλώ την **αναδρομική** συνάρτηση **minPartition(int[] S, int n, int S1, int S2, Map<String, Integer > lookup)** όπου **S** είναι το είναι πίνακας με κάθε κελί να έχει το άθροισμα αδειών ενός φύλακα, **n** το πλήθος των αθροισμάτων, τα **S1** και **S2** είναι το άθροισμα κάθε συνόλου αντίστοιχα και το **lookup** είναι ένα **map** με μοναδικά κλειδιά και δυναμικό **input** από τα στοιχεία του **S**. Όταν η **minPartition** μπει σε κατάσταση όπου όλα τα στοιχεία του **S** έχουν χωριστεί στα **S1** και **S2** τότε γεμίζω 2 global λίστες τυπου **integer** **S1** και **S2** με τα αθροίσματα αντίστοιχα και μία global λίστα **dif** με τις διαφορές τους. Αυτές οι 3 λίστες γεμίζουν ταυτόχρονα με τον ίδιο δείκτη. Τώρα για κάθε σύνολο του **S1** βρήσκω την αντίστοιχη διαφορά μέχρι να βρω την διαφορά που είναι η ελάχιστη που μου επέστρεψε η **minPartition**. Τώρα ξέρω ποιά αθροίσματα αδειών έχουν την ελάχιστη διαφορά.

Πρέπει να βρω ποιά σύνολα φυλάκων έχουν αυτά τα αθροίσματα αδειών. Η κλάση **Possible\_Worker\_set** έχει μία μέθοδο **printAllSubsets(Integer[] arr, int n, int sum)** η οποία εμφανίζει όλα τα σύνολα που έχουν ένα συγκεκριμένο άθροισμα. Ο **arr** είναι πίνακας με όλα τα αθροίσματα αδειών κάθε φύλακα. Το **n** είναι το μέγιστο

**user\_id** διότι φτιάχνω πίνακες με μέγεθος  $\text{max\_user\_id}+1$  και το **neededSum** είναι το άθροισμα για το οποίο θέλω να βρώ **subset**. Μετά το τέλος της **printAllSubsets** έχω σύνολο με τα **id** των φυλάκων που οι άδειές τους έχουν άθροισμα το ζητούμενο άθροισμα. Παίρνω το συμπληρωματικό σύνολο στο πρώτο και έτσι έχω χωρίσει τους εργαζόμενους με άδειες σε 2 σύνολα. Μετά αυτά τα 2 σύνολα τα γεμίζω εξίσου με τους υπόλοιπους φύλακες που δεν έχουν άδειες.

Τέλος κατασκευάζω τα 2 mapping των 2 συνόλων. Είναι λίστες τύπου **employeeMapping** όπου το **index** είναι στο εύρος [0 - πλήθος εργαζομένων για το συγκεκριμένο σύνολο) και το περιεχόμενο είναι το **user\_id** και το **index**. Επίσης ετοιμάζω πόσους εργαζόμενους θα έχει το κάθε κτίριο και για κάθε εργαζόμενο ετοιμάζω έναν πίνακα με τις βάρδιες που έκανε σε αργίες.

Ουσιαστικά παίρνω τις ημερομηνίες της εβδομάδας που βγάζω το πρόγραμμα. Παίρνω τις ημέρες άδειας των εργαζομένων όταν η εβδομάδα προγράμματος έχει κοινές ημέρες με τις άδειες. Μετά χωρίζω τους εργαζόμενους με άδειες σε 2 σύνολα που έχουν την ελάχιστη δυνατή διαφορά από τα σύνολα των αδειών. Μετά γεμίζω τα σύνολα με τους υπόλοιπους εργαζόμενους.

#### 4.5.2. Ανάθεση φυλάκων με ευθύνη του admin

Ο χωρισμός των εργαζομένων από τον admin γίνεται με την χρήση **combobox** που επιτρέπουν την επιλογή του κτιρίου. Μόλις πατηθεί το κουμπί **submit** εκτελείται η αντίστοιχη συνάρτηση που ακούει σε αυτό το κουμπί. Αρχικά ελέγχω εάν όλοι οι φύλακες έχουν οριστεί ένα κτίριο ελέγχοντας τα αν ο πίνακας μεγέθους 13 έχει σε όλα τα κελιά του κάποια επιλογή από το **combobox**. Η επιλογές των **combobox** αποθηκεύονται σε έναν **global** πίνακα και κάθε φορά που αλλάζει η τιμή του **combobox** το κελί που αντιστοιχεί στον εργαζόμενο σύμφωνα με την θέση του στον πίνακα αλλάζει. Για παράδειγμα το πρώτο κελί αντιστοιχεί στον πρώτο εργαζόμενο που φαίνεται στην σελίδα. Όταν οριστεί κτίριο για αυτόν το εργαζόμενο γεμίζει το κελί με τον τύπο του κτιρίου. Μετά ελέγχω αν όλα τα κελιά του πίνακα έχουν κάποιο κτίριο.

Παίρνω τον αριθμό εβδομάδας από τον **scheduled\_week** πίνακα και τον αυξάνω. Μετά φτιάχνω τα mappings όπως και στον αυτόματο χωρισμό φυλάκων. Ετοιμάζω και τις συνολικές αργίες των εργαζομένων όπως πριν και επιπλέον καλώ την συνάρτηση **totalVacDays** διότι η συνάρτηση γεμίζει την λίστα **tuvd** από **TotalUserVacDays** που χρειάζομαι για να ξέρω ποιές μέρες έχει ο φύλακας άδεια για να βγάλω σωστά το εβδομαδιαίο πρόγραμμα.

#### 4.5.3. Δημιουργία προγράμματος

Η κλάση που εκδίδει το εβδομαδιαίο πρόγραμμα έχει μία κύρια συνάρτηση που ονομάζεται **schedule( int call)**. Η μεταβλητή **call** καθορίζει για ποιο κτίριο βγάζω το πρόγραμμα, **call=1** είναι για το κτίριο των Βουτών, **call=2** είναι για το κτίριο Κνωσσού. Ελέγχοντας την **call** μεταβλητή ελέγχω ποιές μεταβλητές και **mapping** χρησιμοποιώ για τα **uid** και πόσο είναι το πλήθος των εργαζομένων για το συγκεκριμένο κτίριο.

Αρχικά εκτελώ ένα **query select** στο πίνακα **user** για να βρω τους 3 εργαζόμενους με τις περισσότερες βάρδιες αργίας και τους 3 λιγότερους. Αποθηκεύω τα **uid** σε **global** μεταβλητές.

Μετά δημιουργώ ένα αντικείμενο τύπου **ShiftSchedule**. Αρχικοποιώ την λίστα τύπου **Shift** με καινούργιο αντικείμενο τύπου **Shift** και αναθέτω σε αυτό το αντικείμενο τον αριθμό βάρδιας όπου ανήκει. Δημιουργώ 20 τέτοια **Shift** αντικείμενα. Για κάθε τέτοιο **Shift** object δημιουργώ 3 αντικείμενα τύπου **ShiftAssignment** και τα εισάγω στην λίστα **shiftListAssignment** του **ShiftSchedule** και αρχικοποιώ κάθε αντικείμενο **ShiftAssignment** με ένα τον αριθμό βάρδιας όπου ανήκουν. Έτσι ξέρω ποιά 3 αντικείμενα στη **shiftListAssignment** ανήκουν στη βάρδια που ελέγχω και έτσι ξέρω ποιοί εργαζόμενοι ανήκουν στην βάρδια. Μετά εισάγω και μία λίστα στο **ShiftSchedule** τύπου **Integer** που αντιπροσωπεύει το εύρος των **id** των εργαζομένων ως συνεχόμενους αριθμούς. Το εύρος το εξάγω από το πλήθος των εργαζομένων. Το εύρος είναι συνεχόμενοι αριθμοί [0 - πλήθος εργαζομένων κτιρίου) έτσι ώστε να μπορώ να ορίζω πίνακες όπου οι δείκτες αντιπροσωπεύουν το **id** εργαζόμενου. Για παράδειγμα ένας πίνακας 13x7 δείχνει ποιοί εργαζόμενοι εργάζονται ποιές ημέρες. Χρησιμοποιώντας τα **mapping** που έφτιαξα στον χωρισμό των εργαζομένων μπορώ να αντιστοιχίσω τον δείκτη αριθμό στο πραγματικό **id** του εργαζόμενου.

Αφού αρχικοποιήσω την οντότητα **ShiftSchedule** για το Optaplanner καλώ τον **solver** και σύμφωνα με κανόνες που του έχω εισάγει στη κλάση **ScoreCalculator implements EasyScoreCalculator<ShiftSchedule>** βαθμολογεί τις λύσεις που βγάζει αναθέτοντας εργαζόμενους στον **ShiftAssignment** που εξάγει από το **ShiftSchedule**. Για κάθε βάρδια δηλαδή εισάγει εργαζόμενο. Γεμίζει όλη την λίστα **shiftListAssignment** με εργαζόμενους στη κάθε βάρδια. Μετά βαθμολογεί αυτή τη λίστα ως λύση του προβλήματος σύμφωνα με κανόνες που όρισα εγώ. Στο τέλος επιστρέφει την “καλύτερη” λύση.

#### 4.5.3.1. Δημιουργία seed προγράμματος

Η μέθοδος με τους κανόνες που βαθμολογεί ονομάζεται **Score calculateScore(ShiftSchedule shiftSchedule)**. Αρχικά δημιουργώ έναν πίνακα πλήθος εργαζομένων  $x$  3 για να ελέγξω ποιοι εργαζόμενοι μπήκαν στην συγκεκριμένη ημέρα και σε ποιά βάρδια. Μετα φτιάχνω έναν πίνακα για να μετράω τις συνολικές βάρδιες κάθε εργαζόμενου (μονοδιάστατος μεγέθους το πλήθος των εργαζομένων) και έναν πίνακα για το σύνολο των εβδομαδιαίων βαρδιών του εργαζόμενου (μονοδιάστατος μεγέθους το πλήθος των εργαζομένων) , και έναν πίνακα για τον έλεγχο συνεχόμενων νυχτερινών βαρδιών του εργαζόμενου (δισδιάστατος  $21 \times$  πλήθος εργαζομένων) .

Για κάθε **ShiftAssignment** στο **shiftSchedule.getShiftListAssignment()** ελέγχω αν έχει οριστεί εργαζόμενος. Αν έχει οριστεί, τότε παίρνω από το ShiftAssignment την βάρδια και τον εργαζόμενο. Αρχικά ελέγχω τον πίνακα εργαζομένων  $x$  βαρδιών αν ο εργαζόμενος είναι είδη στην βάρδια. Αν ναι τότε ρίχνω το score της λύσης αλλιώς αυξάνω την τιμή του πίνακα στο κελί αυτό ( 0 -> 1) . Αν η βάρδια είναι βραδινή αυξάνω το κελί του πίνακα με τις νυχτερινές βάρδιες. Μετα αυξάνω το κελί του εργαζόμενου στον πίνακα των ημερήσιων βαρδιών και μετά το κελί του πίνακα των εβδομαδιαίων βαρδιών. Μετά ελέγχω για αυτόν τον εργαζόμενο αν έχει περισσότερες από 5 βάρδιες την εβδομάδα. Αν έχει ακριβώς 5 ανεβάζω το score της λύσης, αλλιως το χαμηλώνω. Ελέγχω αν έχει παραπάνω από μία βάρδια την ημέρα. Αν έχει τότε χαμηλώνω το score της λύσης. Στη συνέχεια ελέγχω αν ο εργαζόμενος είναι ένας από τους εργαζόμενους με τις περισσότερες βάρδιες σε αργίες. Ελέγχω αν η βάρδια ανήκει σε αργία και αν ναι τότε χαμηλώνω το score της λύσης. Αν δεν έχει οριστεί εργαζόμενος στην βάρδια ελέγχω τον πίνακα με τις βάρδιες και τους εργαζόμενους και ελέγχω αν η βάρδια έχει τουλάχιστον ένα εργαζόμενο. Αν όχι τότε μειώνω το score της λύσης. Το Optaplanner εργαλείο δεν δημιουργεί μία λύση που τηρεί εξ ολοκλήρου τους κανόνες που του ορίζονται, αλλά δημιουργεί μία προσεγγιστική λύση.

#### 4.5.3.2. Αντιγραφή για μορφοποίηση του seed

Αφού ολοκληρώσει η **calculateScore** επιστρέφω στην **schedule** μέθοδο. Δημιουργώ μία λίστα τύπου **finalShift** που κρατάει δύο Integers για το id εργαζόμενου και την βάρδια. Παίρνω την λύση που μου επιστρέφει το Optaplanner και για κάθε βάρδια αντιγράφω το id του εργαζόμενου σε αντικείμενο τύπου **finalShift** και το εισάγω στην λίστα. Αυτή η λίστα είναι η τελική λίστα με το πρόγραμμα όπου θα κάνω όλες τις απαραίτητες αλλαγές για να φτιάξω ένα νόμιμο πρόγραμμα. Οι βάρδιες και τα id μπαίνουν με την σειρά στην λίστα. Οπότε ξέρω τα πρώτα 3 αντικείμενα της λίστας

αντιστοιχούν στην πρώτη βάρδια. Τα πρώτα 9 αντικείμενα της λίστας αντιστοιχούν στην πρώτη ημέρα της λίστας.

Το Optaplanner γεμίζει με πολλά duplicates την λύση. Πρέπει να την μορφοποιήσω κατάλληλα.

#### **4.5.3.3. Αφαίρεση επαναλαμβανόμενων καταχωρίσεων εργαζομένων στην ίδια βάρδια.**

Αρχικά μετράω το σύνολο των βαρδιών κάθε εργαζόμενου και το εισάγω σε πίνακα όπου ο δείκτης είναι το id του εργαζόμενου. Για κάθε εργαζόμενο σε αυτόν το πίνακα ελέγχω αν ο εργαζόμενος έχει περισσότερες από 5 βάρδιες. Αν ναι, τότε παίρνω το πλήθος των επιπλέον βαρδιών και για κάθε παραπάνω βάρδια ψάχνω τα αντικείμενα της τελικής λίστας ανα τριάδες και ψάχνω αν υπάρχει ο εργαζόμενος. Αν υπάρχει τότε τον αφαιρώ και μειώνω το πλήθος των επιπλέον βαρδιών μέχρι να γίνει 0. Επιπλέον ελέγχω αν κάποιος εργαζόμενος έχει λιγότερες από 5 βάρδιες. Τότε παίρνω το συμπλήρωμα και όπως πριν αντι να αφαιρώ, εισάγω εργαζόμενο σε βάρδια.

Μετά φτιάχνω 3 πίνακες για τις 3 βάρδιες της ημέρας μεγέθους το πλήθος των εργαζομένων. Παίρνω από την τελική λίστα σε εννιάδες τα αντικείμενα και ανα τριάδες γεμίζω τους κατάλληλους πίνακες με το ποιός εργαζόμενος υπάρχει σε αυτή τη βάρδια. Μετά για κάθε εργαζόμενο ελέγχω κάθε πίνακα αν κάποιος εργαζόμενος εμφανίζεται περισσότερο από μία φορά στην ίδια βάρδια. Δηλαδή αν σε μία συνεχόμενη τριάδα βάρδιας υπάρχει πολλαπλές φορές ο ίδιος εργαζόμενος. Αν υπάρχει τότε τον αφαιρώ μία ή δύο φορές, μέχρι να υπάρχει μία φορά. Μετά ελέγχω τους πίνακες σε ζευγάρια και ελέγχω αν ο στον ίδιο δείκτη ( ίδιος εργαζόμενος) υπάρχει και στους 2 πίνακες η μονάδα. Έτσι ξέρω ότι ο εργαζόμενος εμφανίζεται σε παραπάνω από μία βάρδια. Αν ο εργαζόμενος εμφανίζεται στην πρωινή και μεσημεριανή βάρδια τότε βγάζω τον εργαζόμενο από μεσημεριανή βάρδια. Αν πρωινή και βραδινή, ελευθερώνω την βραδινή. Αν μεσημεριανή και βραδινή, ελευθερώνω βραδινή.

Τώρα κάθε εργαζόμενος ανήκει σε μία βάρδια την ημέρα και εμφανίζεται μία φορά σε αυτή τη βάρδια.

#### **4.5.3.4. Έλεγχος βραδινών βαρδιών**

Τώρα ελέγχω τις βραδινές βάρδιες. Φτιάχνω ένα πίνακα μεγέθους το πλήθος των εργαζομένων. Και για κάθε βάρδια ελέγχω αν είναι βραδινή και αυξάνω τον μετρητή του εργαζομένου. Μετράω πόσες βραδινές βάρδιες έχει ο κάθε εργαζόμενος. Βρίσκω ποιός εργαζόμενος έχει τις περισσότερες και ποιος τις λιγότερες βραδινές βάρδιες.

Όσο υπάρχει εργαζόμενος με περισσότερες από 3 βραδινές βάρδιες και εργαζόμενος με λιγότερες από 3 βραδινές ψάχνω να βρώ βραδινή βάρδια με τον εργαζόμενο με τις επιπλέον βραδινές βάρδιες και ψάχνω πρωινή ή μεσημεριανή βάρδια με τον εργαζόμενο με τις λιγότερο από 3 βραδινές βάρδιες έτσι ώστε να τους αλλάξω θέση στη λίστα. Πρέπει πρώτα όμως να ελέγξω αν η ημέρα που πάω να βάλω τον καινούργιο εργαζόμενο για βράδυ δεν τον έχει ήδη σε μία άλλη βάρδια της ημέρας. Δηλαδή να μην παραβώ την μία βάρδια ανα ημέρα. Αυτό το κάνω πέρνωντας ανα εννιάδες την τελική λίστα προγράμματος, σε πίνακα μεγέθους πλήθος εργαζομένων για κάθε id που βρίσκω το χρησιμοποιώ ως δείκτη και αυξάνω το περιεχόμενο του πίνακα. Στο τέλος της εννεάδας ελέγχω αν ο εργαζόμενος υπάρχει η όχι σε αυτή την εννιάδα-ημερα. Αν όχι τότε μπορώ να τον βάλω. Στη συνέχεια ξαναβρήσκω τους max και min εργαζόμενους με βραδινές βάρδιες. Και όσο συμβαίνει αυτό προσπαθώ να κάνω αλλαγές μέχρι να μην μπορώ να μετακινήσω ή κάποια αλλαγή να μην γίνεται, και απλώς σταματάω να ψάχνω για αλλαγές.

Στη συνέχεια χωρίζω τις βάρδιες σε τριάδες. Για κάθε βάρδια στην τριάδα γεμίζω πίνακες μεγέθους το πλήθος των εργαζομένων με τους εργαζόμενους για την βραδινή, την επόμενη πρωινή και την επόμενη μεσημεριανή βάρδια. Ελέγχω ποιοί και πόσοι εργαζόμενοι εργάζονται βράδυ και το επόμενο πρωί. Ελέγχω ποιοί και πόσοι δεν εργάζονται βράδυ και εργάζονται το επόμενο μεσημέρι. Αφού κάθε εργαζόμενος εμφανίζεται μία φορά στην συγκεκριμένη ημέρα, αν κάποιος εργάζεται βράδυ και επόμενο πρωί και ένα εργαζόμενος την επόμενης ημέρας εργάζεται μεσημέρι και όχι το προηγούμενο βράδυ μπορώ να τους αλλάξω βάρδια στην ίδια ημέρα. Έτσι ο εργαζόμενος που εργάζεται το βράδυ, η επόμενη βάρδια του είναι μεσημέρι, και αυτός που δεν εργάζεται βράδυ μπορεί να εργαστεί το επόμενο πρωί. Επιπλέον σε περίπτωση που δεν υπάρχει μεσημεριανός εργαζόμενος που να μπορεί να μπει πρωί ελέγχω αν η μεσημεριανή βάρδια έχει κενό, δηλαδή λιγότερο από 3 εργαζόμενους. Αν έχει κενό τότε απλώς μεταφέρω την πρωινή βάρδια σε μεσημεριανή.

#### **4.5.3.5. Γέμισμα άδειων βαρδιών**

Στη συνέχεια φτιάχνω ένα πίνακα μεγέθους 21 που αντιπροσωπεύει τις βάρδιες και σημειώνω ποιές βάρδιες είναι άδειες. Φτιάχνω ένα πίνακα μεγέθους το πλήθος των εργαζομένων και σημειώνω την διαφορά 5-συνολικές βάρδιες εργαζομένου. Για κάθε άδεια βάρδια φτιάχνω 3 πίνακες μεγέθους πλήθος εργαζομένων που κρατάει πρώτος τους εργαζόμενους της προηγούμενης βραδιάς, ο δεύτερος τους εργαζόμενους της ολικής ημέρας όπου ανήκει η άδεια βάρδια και ο τρίτος τους εργαζόμενους του επόμενου πρωί. Χρησιμοποιώ τους 3 πίνακες για ελέγχους. Αν η άδεια βάρδια είναι πρωινή ελέγχω αν ο κάποιος εργαζόμενος έχει λιγότερες από 5 βάρδιες και δεν εργάζεται το προηγούμενο βράδυ και δεν εργάζεται όλη την επόμενη

ημέρα με την άδεια βάρδια. Αν ισχύει αυτό τότε εισάγω αυτόν τον εργαζόμενο στη πρωινή βάρδια και μειώνω την διαφορά κατά 1 και τις άδειες βάρδιες κατά 1. Αν η άδεια βάρδια είναι μεσημέρι αρκεί να βρώ εργαζόμενο με λιγότερες από 5 βάρδιες και να μην εργάζεται την συγκεκριμένη ημέρα. Αν ναι τότε τον εισάγω. Αν η βάρδια είναι βραδινή ελέγχω αν υπάρχει κάποιος εργαζόμενος που δεν εργάζεται όλη μέρα ούτε το επόμενο πρωί και έχει λιγότερες βάρδιες, τότε τον εισάγω.

Στη συνέχεια δημιουργώ ξανά τους πίνακες για το προηγούμενο βράδυ, σημερινή ημέρα και επόμενο βράδυ όπως πριν, και ελέγχω αν υπάρχει εργαζόμενος που δεν έχει συμπληρώσει τις απαραίτητες εβδομαδιαίες βάρδιες και ψάχνω μία κατάλληλη βάρδια για να τον εισάγω.

#### 4.5.3.6. Έλεγχος αδειών εργαζομένων

Τώρα πρέπει να ελέγξω για άδειες εργαζομένων. Στον **schedule\_creator\_Controller** όταν έφτιαχνα και χώριζα τους εργαζόμενους σε ομάδες έφτιαχνα μία λίστα **TotalUserVacDays** που κρατάει το πλήθος των αδειών ενός εργαζομένου. Στον **ScheduleGenerator** υπάρχει μία μέθοδος που παίρνει αυτή την λίστα και μέσω του mapping για τα ids των εργαζομένων γεμίζει έναν πίνακα 2 διαστάσεων μεγέθους πλήθος εργαζομένων X 7 ημέρες έτσι ώστε να ξέρω ποιος εργαζόμενος έχει άδεια ποιά ημέρα. Με το index που αντιστοιχεί στην ημέρα από αυτόν τον πίνακα μπορώ να βρώ ποιά στοιχεία της τελικής λίστας με τις βάρδιες αντιστοιχούν σε αυτή την ημέρα. Ελέγχω αυτά τα 9 αντικείμενα της λίστας και βλέπω αν υπάρχει ο εργαζόμενος. Αν ναι τότε τον διαγράφω. Επιπλέον αν έχει ρεπό την ημέρα της άδειας τότε ψάχνω μία άλλη βάρδια όπου εργάζεται για να του δώσω ρεπό.

Μετά ψάχνω ξανά αν υπάρχει άδεια βάρδια λόγω των αδειών. Για κάθε άδεια βάρδια ελέγχω ποιά βάρδια έχει τουλάχιστον 2 εργαζόμενους και μπορεί να εργαστεί αυτή την άδεια βάρδια για να τον μεταφέρω τότε. Φτιάχνω ξανά τους 3 πίνακες προηγούμενο βράδυ, σημερινή ημέρα και επόμενο πρωί. Ελέγχω την ημέρα όπου ανήκει η άδεια βάρδια να έχω τα στοιχεία των εργαζομένων για το προηγούμενο βράδυ, σημερινή ημέρα και επόμενο πρωί ανάλογα με τον τύπο της βάρδιας, αν είναι πρωί, μεσημέρι ή βράδυ. Εάν η άδεια βάρδια είναι μεσημέρι τότε απλώς μεταφέρω τον εργαζόμενο. Αν είναι πρωί τότε κοιτάζω αν εργάστηκε το προηγούμενο βράδυ. Αν είναι βράδυ ελέγχω αν εργάζεται το επόμενο πρωί και αν έχει λιγότερες από 3 βραδινές βάρδιες ήδη. Για κάθε βάρδια ελέγχω αν ο εργαζόμενος έχει άδεια εκείνη την ημέρα. Επιπλέον ελέγχω αν η βάρδια στην οποία ανήκει ο εργαζόμενος που πάει να μετακινησω είναι ο μόνος εργαζόμενος αυτής της βάρδιας. Υπάρχει μέθοδος ονόματι **soloEmployee** που παίρνει τη θέση της βάρδιας στη λίστα και την λίστα με το πρόγραμμα και βλέπω σύμφωνα με την θέση του index στη λίστα ανάλογα με την

θέση του στη βάρδια αν τα υπόλοιπα 2 αντικείμενα της βάρδιας έχουν εργαζόμενο ή όχι. Index αντιστοιχεί σε slot της βάρδιας όπου σίγουρα υπάρχει ο εργαζόμενος που θέλω να μετακινήσω.

Τέλος κάνω έναν τελευταίο έλεγχο στις συνολικές βάρδιες των εργαζομένων για να είμαι σίγουρος ότι ολοκληρώνουν όλες τις υποχρεωτικές τους βάρδιες σύμφωνα με τις άδειες που έχουν ζητήσει. Και αν υπάρχει κάποιος με λιγότερες βάρδιες από όσες πρέπει τότε ψάχνω μία κατάλληλη βάρδια σύμφωνα με τις ημέρες άδειας και πλήθος βραδινών βαρδιών και τον εισάγω.

Τώρα έχω μία λίστα με το τελικό πρόγραμμα έτοιμο για αποθήκευση στον βάση. Καλώ την συνάρτηση **storeSchedule** που παίρνει σε τριάδες τους εργαζόμενους της βάρδιας, την εβδομάδα για την ημέρα της βάρδιας, τον τύπο της κλήσης της συνάρτησης για το ποιο κτίριο εισάγω τις βάρδιες. Το id που παίρνει η συνάρτηση είναι το πραγματικό id του εργαζομένου. Έχει γίνει αντιστοίχιση **index-id** πριν την κλήση της συνάρτησης. Έχω μια συνάρτηση που δημιουργεί και εισαγει τα δεδομένα σε αντικείμενα τύπου **calendarevent** και τα επιστρέφει. Για κάθε εργαζόμενο που δεν είναι null κάνω ένα **session.save(calendarevent object)** και αποθηκεύω την βάρδια.



## 5. Αλλαγές, Βελτιώσεις, Επιπλέον features για μελλοντικό development

Λόγω περιορισμένης εμπειρίας υπάρχουν πολλές προγραμματιστικές επιλογές που έγιναν κατά την δημιουργία της εφαρμογής που σε μελλοντική έκδοση θα διορθώνονταν.

Αρχικά ο τρόπος επικοινωνίας των σελίδων frontend και της κλάσης controller θα γίνονταν με πιο σωστή αντικειμενοστρέφια και λιγότερες global μεταβλητές.

Για την έκδοση του προγράμματος δεν θα χρησιμοποιούσα το εργαλείο OrtaPlanner και θα έφτιαχνα το πρόγραμμα έτσι ώστε να έχω την ελευθερία να κάνω αρχικοποίηση το πρόγραμμα με δεδομένα του χρήστη όπως:

- Εισαγωγή εργαζομένων σε συγκεκριμένες βάρδιες.
- Ορισμό ελαχίστου ορίου πλήθους εργαζομένων αν τύπο βάρδιας ή συγκεκριμένης βάρδιας συγκεκριμένης ημέρας.
- Δυνατότητα ορισμού προτεραιότητας σε βάρδιες για την ανάθεση εργαζομένων.

Αυτόματη μορφοποίηση του προγράμματος μετά την έκδοση του στην περίπτωση που έχει δοθεί άδεια σε εργαζόμενο έτσι ώστε να γεμίσουν τυχόν κενά στο πρόγραμμα. Αυτή τη στιγμή αν δοθεί άδεια σε εργαζόμενο σε εβδομάδα όπου έχει εκδοθεί το πρόγραμμα δεν γίνεται αυτόματα η αφαίρεση του εργαζομένου. Πρέπει να την σβήσει ο admin την βάρδια.

Η υπάρχουσα έκδοση του χρονοπρογραμματιστή δημιουργεί μία βασική έκδοση προγράμματος όπου προσέχει οι εργαζόμενοι που ορίζει τις βάρδιες να μπορούν να εργαστούν. Να μην είναι δηλαδή παράνομες βάρδιες, είτε λόγω άδειας ή βραδινής βάρδιας. Κάθε εργαζόμενος

εργάζεται μία φορά την ημέρα, πέντε φορές την εβδομάδα. Εάν εργάζεται βράδυ τότε δεν εργάζεται το επόμενο πρωί. Εάν έχει άδεια τότε δεν εμφανίζεται στο πρόγραμμα της ημέρα εκείνης.

Έγινε μία προσπάθεια με το Optaplanner να γίνει δίκαιο το πρόγραμμα ως προς τις αργίες κοιτάζοντας το πλήθος Κυριακών που εργάστηκε ο εργαζόμενος και να δώσω βαρύτητα σε εκείνους με τις λιγότερες βάρδιες σε αργίες αλλά δεν δουλεύει σωστά.

Η εφαρμογή αποθηκεύει αυτόματα τις ημερήσιες βάρδιες και μπορεί να ελέγχει αν είναι αργία ή νυχτερινή έτσι ώστε να μπορεί να εξάγει ο χρήστης την σωστή μισθοδοσία. Υπάρχουν οι αργίες του δημοσίου ορισμένες στον κώδικα.

#### **5.1. Συμπεριφορά χρονοπρογραμματιστή σε ακραίες περιπτώσεις**

1. Καθώς ο χρονοπρογραμματιστής επιτρέπει την ύπαρξη κενών βαρδιών στην περίπτωση που δεν υπάρχουν αρκετοί εργαζόμενοι τότε η βάρδια αφήνεται κενή.
2. Ένα πρόβλημα που δεν πρόλαβα να λύσω είναι ότι δεν ελέγχεται η νομιμότητα της πρώτης βάρδιας της εβδομάδας σύμφωνα με το αν ο εργαζόμενος εργάστηκε το προηγούμενο βράδυ του προηγούμενου προγράμματος.
3. Εάν έχει πολλούς εργαζόμενους και το πρόγραμμα είναι γεμάτο και δεν υπάρχουν slots για να μπει εργαζόμενος τότε αυτός ο εργαζόμενος μπορεί να έχει λιγότερες από 5 βάρδιες. Το μέγιστο 3 εργαζόμενοι ανα βάρδια είναι hardcoded. Στο μέλλον θα το άλλαζα να είναι ελαστικό και στο μέγιστο πλήθος εργαζομένων σε βάρδια, πέραν του ελαχίστου.

## 6. References

- JAVA SE docs  
<https://docs.oracle.com/javase/8/docs/>
- Eclipse  
<https://www.eclipse.org/documentation/>
- ZK Studio  
<https://www.zkoss.org/documentation/zkstudio>
- Maven  
<https://maven.apache.org/what-is-maven.html>
- MySQL 8.0 and Workbench  
<https://dev.mysql.com/doc/refman/8.0/en/>  
<https://dev.mysql.com/doc/workbench/en/>
- Hibernate ORM  
<https://hibernate.org/orm/>
- Quartz Scheduler  
<http://www.quartz-scheduler.org/documentation/>
- OptaPlanner  
[https://docs.optaplanner.org/7.39.0.Final/optaplanner-docs/html\\_single/index.html](https://docs.optaplanner.org/7.39.0.Final/optaplanner-docs/html_single/index.html)